Hadoop HDFS本地存储目录结构解析

HDFS metadata以树状结构存储整个HDFS上的文件和目录，以及相应的权限、配额和副本因子（replication factor）等。本文基于Hadoop2.6版本介绍HDFS Namenode本地目录的存储结构和Datanode数据块存储目录结构，也就是hdfs-site.xml中配置的dfs.namenode.name.dir和dfs.namenode.data.dir。

一、NameNode

HDFS metadata主要存储两种类型的文件

1、fsimage

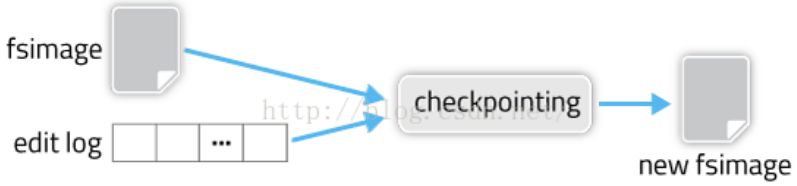
记录某一永久性检查点（Checkpoint）时整个HDFS的元信息

2、edits

所有对HDFS的写操作都会记录在此文件中

Checkpoint介绍

HDFS会定期（dfs.namenode.checkpoint.period，默认3600秒）的对最近的fsimage和一批新edits文件进行Checkpoint（也可以手工命令方式），Checkpoint发生后会将前一次Checkpoint后的所有edits文件合并到新的fsimage中，HDFS会保存最近两次checkpoint的fsimage。Namenode启动时会把最新的fsimage加载到内存中。



下面是一个标准的dfs.namenode.name.dir目录结构，注意edits和fsimage也可以通过配置放到不同目录中

```
[html]
```

```
├── current
│   ├── VERSION
│   ├── edits_0000000000000000001-0000000000000000007
│   ├── edits_0000000000000000008-0000000000000000015
│   ├── edits_0000000000000000016-0000000000000000022
│   ├── edits_0000000000000000023-0000000000000000029
│   ├── edits_0000000000000000030-0000000000000000030
│   ├── edits_0000000000000000031-0000000000000000031
│   ├── edits_inprogress_0000000000000000032
│   ├── fsimage_0000000000000000030
│   ├── fsimage_0000000000000000030.md5
│   ├── fsimage_0000000000000000031
│   ├── fsimage_0000000000000000031.md5
│   └── seen_txid
└── in_use.lock
```

1、VERSION

```
[html]
#Thu May 19 10:13:22 CST 2016
namespaceID=1242163293
clusterID=CID-124668a8-9b25-4ca7-97bf-5dd5c25041a9
cTime=1455091012961
storageType=NAME_NODE
blockpoolID=BP-180412957-192.168.1.8-1419305031110
layoutVersion=-60
```

- layoutVersion - HDFS metadata版本号，通常只有HDFS增加新特性时才会更新这个版本号
- namespaceID/clusterID/blockpoolID - 这三个ID在整个HDFS集群全局唯一，作用是引导Datanode加入同一个集群。在HDFS Federation机制下，会有多个Namenode，所以不同Namenode直接namespaceID是不同的，分别管理一组blockpoolID，但是整个集群中，clusterID是唯一的，每次format namenode会生成一个新的，也可以使用-clusterid手工指定ID
- storageType - 有两种取值NAME_NODE /JOURNAL_NODE，对于JournalNode的参数dfs.journalnode.edits.dir，其下的VERSION文件显示的是JOURNAL_NODE
- cTime - HDFS创建时间，在升级后会更新该值

2、edits_start transaction ID-end transaction ID

finalized edit log segments，在HA环境中，Standby Namenode只能读取finalized log segments，

3、edits_inprogress__start transaction ID

当前正在被追加的edit log，HDFS默认会为该文件提前申请1MB空间以提升性能

4、fsimage_end transaction ID

每次checkpoing（合并所有edits到一个fsimage的过程）产生的最终的fsimage，同时会生成一个.md5的文件用来对文件做完整性校验

5、seen_txid

保存最近一次fsimage或者edits_inprogress的transaction ID。需要注意的是，这并不是Namenode当前最新的transaction ID，该文件只有在checkpoing(merge of edits into a fsimage)或者edit log roll(finalization of current edits_inprogress and creation of a new one)时才会被更新。

这个文件的目的在于判断在Namenode启动过程中是否有丢失的edits，由于edits和fsimage可以配置在不同目录，如果edits目录被意外删除了，最近一次checkpoint后的所有edits也就丢失了，导致Namenode状态并不是最新的，为了防止这种情况发生，Namenode启动时会检查seen_txid，如果无法加载到最新的transactions，Namenode进程将不会完成启动以保护数据一致性。

6、in_use.lock

防止一台机器同时启动多个Namenode进程导致目录数据不一致

二、Datanode

Datanode主要存储数据，下面是一个标准的dfs.datanode.data.dir目录结构

```
[html]
01. ├── current
02. │ ├── BP-1079595417-192.168.2.45-1412613236271
03. │ │ ├── current
04. │ │ │ ├── VERSION
05. │ │ │ ├── finalized
06. │ │ │ │ └── subdir0
07. │ │ │ │ └── subdir1
08. │ │ │ │ ├── blk_1073741825
09. │ │ │ │ └── blk_1073741825_1001.meta
10. │ │ │ ├── lazyPersist
11. │ │ │ └── rbw
12. │ │ ├── dncp_block_verification.log.curr
13. │ │ ├── dncp_block_verification.log.prev
14. │ │ └── tmp
15. │ └── VERSION
16. └── in_use.lock
```

1、BP-random integer-NameNode-IP address-creation time

BP代表BlockPool的意思，就是上面Namenode的VERSION中的集群唯一blockpoolID，如果是Federation HDFS，则该目录下有两个BP开头的目录，IP部分和时间戳代表创建该BP的NameNode的IP地址和创建时间戳

2、VERSION

与Namenode类似，其中storageType是DATA_NODE

```
[html]
01. #Wed Feb 10 16:00:18 CST 2016
02. storageID=DS-2e165f84-68b1-40c9-b501-b6b08fcb09ee
03. clusterID=CID-124668a8-9b25-4ca7-97bf-5dd5c25041a9
04. cTime=0
05. datanodeUuid=cb9fead7-cd64-4507-affd-c06f083708b5
06. storageType=DATA_NODE
07. layoutVersion=-56
```

3、finalized/rbw目录

这两个目录都是用于实际存储HDFS BLOCK的数据，里面包含许多block_xx文件以及相应的.meta文件，.meta文件包含了checksum信息。

rbw是“replica being written”的意思，该目录用于存储用户当前正在写入的数据。

4、dncp_block_verification.log

该文件用于追踪每个block最后修改后的checksum值，该文件会定期滚动，滚动后会移到.prev文件

5、in_use.lock

防止一台机器同时启动多个Datanode进程导致目录数据不一致